Visualización, partición y automatización en el procesamiento de datos

Breve descripción:

Este componente aborda el ciclo de vida del aprendizaje automático, incluyendo la limpieza, transformación y evaluación de datos. Presenta técnicas de modelado, automatización de procesos, visualización de resultados y buenas prácticas para la gestión eficiente de datos en proyectos de inteligencia artificial.

Julio 2025

Tabla de contenido

Introducción

La gestión del ciclo de vida del aprendizaje automático abarca todas las fases necesarias para desarrollar soluciones basadas en inteligencia artificial de forma eficiente y sostenible. Este proceso incluye la limpieza, transformación y preparación de los datos, así como la selección y evaluación de modelos, la automatización de tareas y la aplicación de buenas prácticas. Una adecuada comprensión de estas etapas permite optimizar los resultados, mejorar la reproducibilidad y reducir errores en la toma de decisiones. Para profundizar en la importancia de estos temas, se recomienda acceder al siguiente video:

Video 1. Visualización, partición y automatización en el procesamiento de datos

[Enlace de reproducción del video]

Síntesis del video: Visualización, partición y automatización en el procesamiento de datos
El ciclo de vida del aprendizaje automático no comienza con el modelo, sino con los datos. Este proceso abarca desde la recolección inicial hasta la evaluación y automatización del flujo de trabajo, permitiendo el desarrollo de soluciones más precisas y confiables. Primero, se realiza una limpieza y transformación de los datos. Esto incluye el tratamiento de valores atípicos, la normalización y la conversión de formatos, tareas fundamentales para asegurar que los algoritmos funcionen correctamente. Posteriormente, los datos se dividen en conjuntos de entrenamiento y prueba, y se seleccionan modelos que serán evaluados con métricas como la precisión o el error cuadrático medio. Una de las prácticas clave en esta etapa es la automatización mediante `pipelines`, como los que ofrece Scikit-learn. Estas estructuras permiten encadenar procesos de forma ordenada y reproducible, garantizando que todos los datos pasen por las mismas etapas de transformación. Asimismo, herramientas como MLflow permiten monitorear experimentos, registrar modelos, evaluar su desempeño y facilitar la toma de decisiones. De manera complementaria, aplicar buenas prácticas de gobernanza de datos asegura calidad, trazabilidad y cumplimiento normativo en todo el proceso. Comprender y dominar este ciclo es esencial para desarrollar modelos eficaces, reducir errores y optimizar el uso de recursos.

1. Preparación de datos en el ciclo de vida del aprendizaje automático

La preparación de datos es una etapa crucial dentro del ciclo de vida del aprendizaje automático, ya que determina la calidad y el rendimiento de los modelos desarrollados. Consiste en un conjunto de técnicas y procesos que permiten convertir datos brutos, desorganizados o incompletos en un insumo confiable, coherente y útil para el entrenamiento de algoritmos. Esta fase no solo mejora la precisión de los modelos, sino que también reduce el riesgo de sesgos, sobreajuste y errores en la toma de decisiones.

En el contexto del aprendizaje automático, los datos son el insumo fundamental para entrenar algoritmos que aprenden patrones y realizan predicciones. Por ello, es indispensable que estén bien preparados antes de iniciar la etapa de modelado. La preparación de datos incluye múltiples actividades como la limpieza, transformación, codificación, normalización, enriquecimiento y tratamiento de valores atípicos. Cada una de estas tareas tiene el objetivo de mejorar la representatividad de los datos y facilitar su procesamiento por parte de los algoritmos.

Esta fase se relaciona estrechamente con otras etapas del ciclo de vida, como la exploración de datos, la selección de características y la evaluación de modelos. Además, una buena preparación no solo incide en la precisión del modelo, sino también en su capacidad para generalizar, lo que es esencial cuando se aplica a nuevos conjuntos de datos en producción.

Una preparación de datos bien ejecutada considera también aspectos éticos y legales, como el tratamiento de datos sensibles, la anonimización de información y la gestión del consentimiento informado, en especial cuando se trabaja con información personal o biomédica.

En resumen, la preparación de datos permite reducir la complejidad del análisis, minimizar errores durante la etapa de entrenamiento y aumentar la confiabilidad de las predicciones. Su correcta ejecución es indispensable para garantizar el éxito de cualquier proyecto de inteligencia artificial o ciencia de datos.

1.1. El ciclo de vida del aprendizaje automático

Según Géron (2020), un proyecto de aprendizaje automático suele desarrollarse a través de una serie de fases estructuradas que permiten transformar datos en modelos útiles para la predicción y la toma de decisiones. Estas etapas son:

a) **Obtención de datos**: consiste en identificar, recopilar y acceder a fuentes de datos relevantes para el problema a resolver. Esto puede implicar el uso de bases de datos públicas, API (Interfaz de Programación de Aplicaciones), archivos locales o flujos en tiempo real.
b) **Partición de los datos**: se divide el conjunto de datos en subconjuntos, comúnmente en un 80 % para entrenamiento y un 20 % para prueba. En algunos casos también se utiliza un conjunto de validación para ajustar hiperparámetros sin sesgar la evaluación final.
c) **Exploración y visualización de los datos**: permite comprender la distribución, relaciones y patrones presentes en los datos. Esta fase ayuda a identificar valores atípicos, datos faltantes y posibles errores.

d) **Preparación y preprocesamiento**: incluye tareas como limpieza, transformación, codificación de variables categóricas (por ejemplo, mediante one-hot encoding), escalado de características numéricas y tratamiento de valores nulos o atípicos. El uso de pipelines facilita la automatización y reproducibilidad de este proceso.

e) **Selección y entrenamiento del modelo**: con base en la naturaleza del problema (clasificación, regresión, clustering, etc.), se elige el algoritmo más adecuado y se entrena con el conjunto de datos preparado.

f) **Evaluación del modelo**: se mide el rendimiento utilizando métricas apropiadas como la precisión, el error cuadrático medio (RMSE), el F1-score, entre otras, dependiendo del tipo de problema.

g) **Ajuste fino (**tuning**) del modelo**: se optimizan los hiperparámetros mediante técnicas como la búsqueda en malla (grid search) o búsqueda aleatoria, con el fin de mejorar la capacidad predictiva del modelo.

h) **Despliegue y mantenimiento**: el modelo entrenado se integra en una aplicación o sistema productivo. Posteriormente, debe ser monitoreado para detectar degradaciones en su rendimiento y actualizarse cuando sea necesario.

Cada una de estas fases es fundamental para asegurar que el modelo sea robusto, generalizable y útil en contextos reales.

1.2. Limpieza y transformación de datos: codificación, normalización y enriquecimiento

La depuración y modificación de datos son etapas esenciales en un proyecto de aprendizaje automático (ML), ya que permiten transformar datos sin procesar en

formatos adecuados para su análisis y modelado. Estos procedimientos comprenden la codificación, la normalización y el enriquecimiento de los datos (Pyle, 1999).

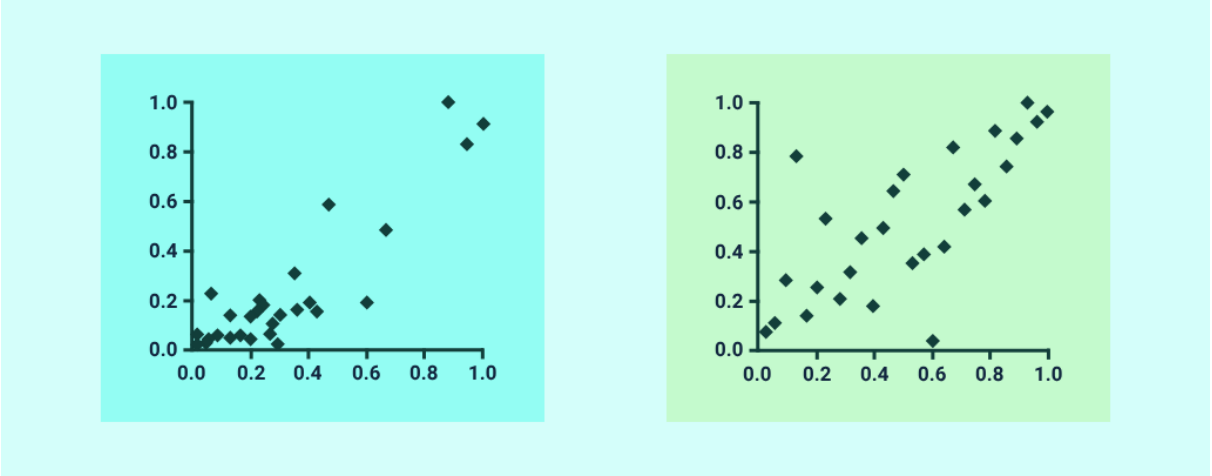
La codificación se refiere a la conversión de variables categóricas en representaciones numéricas. Esta transformación es necesaria, ya que muchos algoritmos de aprendizaje automático requieren datos numéricos para funcionar de manera óptima (Viedma, 2018). Una técnica común es one-hot encoding, que genera columnas binarias para cada categoría, asignando “1” a la categoría presente y “0” a las demás. Esta técnica es especialmente útil cuando se trabaja con variables categóricas nominales (Géron, 2020).

En cuanto a la normalización, Pyle (1999) la define como el proceso de escalar los valores de las variables numéricas a un rango similar. Esta práctica evita que variables con rangos amplios dominen otras durante el entrenamiento de modelos. Una estrategia común es escalar los valores entre 0 y 1. Alternativamente, se puede aplicar la estandarización, ajustando las variables para que tengan una media de cero y una desviación estándar de uno (Géron, 2020).

En redes neuronales profundas, es frecuente utilizar la normalización por lotes (batch normalization), que reparametriza las capas de una red para estabilizar y acelerar el entrenamiento. Esta técnica ajusta las activaciones de cada unidad restando la media y dividiendo entre la desviación estándar del minibatch (Goodfellow et al., 2016).

La siguiente figura, describe cómo diferentes métodos de normalización pueden modificar la representación de los datos y su impacto en el análisis de patrones:

Figura 1. Importancia de normalizar datos



Nota. Tomado de Pyle (1999).

El gráfico izquierdo presenta una normalización simple en rango, lo que reduce la varianza de escala entre atributos, mientras que el gráfico derecho aplica una normalización más profunda, redistribuyendo valores para mejorar la estructura interna de los datos. Esto permite detectar clústeres, definir fronteras y ubicar vecinos más cercanos, lo cual incrementa la eficiencia e interpretabilidad de los algoritmos.

El enriquecimiento de datos, por su parte, consiste en generar nuevas características a partir de las existentes, con el fin de hacer que el conjunto de datos sea más informativo y útil para el modelo. Esta práctica puede mejorar significativamente la precisión y capacidad de generalización (Goodfellow et al., 2016).

En resumen, la transformación de datos incluye la codificación de variables categóricas, la normalización o estandarización de variables numéricas y el enriquecimiento del conjunto de datos mediante la creación de nuevas características. Estos pasos son fundamentales para mejorar el rendimiento de los modelos y pueden automatizarse mediante el uso de `pipelines`.

1.3. Tratamiento de valores atípicos

El tratamiento de valores atípicos, también conocidos como outliers, es una etapa clave en la preparación de datos para modelos de aprendizaje automático. Estos valores, que se alejan significativamente de la tendencia general de los datos, pueden surgir por errores de medición, variabilidad inherente del sistema o por fenómenos excepcionales. Si no se gestionan adecuadamente, pueden distorsionar el entrenamiento del modelo, afectar la precisión de las predicciones y comprometer la validez de los análisis.

El proceso de tratamiento de outliers incluye los siguientes pasos:

1) **Identificación de valores atípicos**: se utilizan técnicas estadísticas y gráficas para detectar datos inusuales, como los diagramas de caja (boxplots), la desviación estándar, el método del rango intercuartílico (IQR) y la puntuación z (z-score).
2) **Análisis contextual**: una vez detectados, es esencial comprender el origen de estos valores. Pueden ser errores de entrada o extremos legítimos que contienen información valiosa para el modelo. El conocimiento del dominio es fundamental para tomar decisiones informadas.
3) **Evaluación del impacto**: se debe analizar cómo los valores atípicos influyen en las métricas del modelo. Por ejemplo, el error cuadrático medio (RMSE) es sensible a valores extremos, lo que puede afectar negativamente la evaluación del rendimiento si no se gestionan adecuadamente.

A partir de esta evaluación, se pueden aplicar diversas estrategias (Pyle, 1999):

- ✓ **Reemplazo**: sustituir los valores atípicos por estimaciones más coherentes, como la media, la mediana o valores imputados por técnicas avanzadas. En algunos casos, se puede añadir ruido blanco para evitar introducir sesgos.
- ✓ **Eliminación**: cuando los outliers se identifican como errores evidentes o irrelevantes para el análisis, es posible eliminarlos. Esta acción debe realizarse con cautela para no perder información valiosa.
- ✓ **Transformación**: aplicar funciones como logaritmos, raíces cuadradas o escalado robusto para reducir el impacto de los valores extremos sin eliminarlos.
- ✓ **Investigación de patrones**: si se detectan múltiples outliers agrupados, puede ser señal de un cambio de comportamiento o un nuevo fenómeno. En tales casos, se recomienda investigar el origen de estas anomalías antes de tomar acciones correctivas.

En conclusión, el adecuado tratamiento de valores atípicos mejora la calidad de los datos y, por tanto, la fiabilidad de los modelos predictivos. Junto con las técnicas de limpieza y transformación, esta tarea fortalece la base sobre la cual se construyen análisis precisos, robustos y útiles en el contexto del aprendizaje automático.

2. Técnicas de partición y balanceo de datos para el modelado

Una vez que los datos han sido limpiados, transformados y enriquecidos, el siguiente paso consiste en organizar el conjunto de datos de manera que permita construir y evaluar modelos de aprendizaje automático de forma confiable. Esta organización implica dividir los datos en subconjuntos con funciones específicas, lo que permite entrenar el modelo, ajustar sus parámetros y evaluar su capacidad de generalización.

Según Géron (2020), las principales estrategias de partición son:

1) **División en conjuntos de entrenamiento y prueba**: la práctica más común consiste en separar el conjunto de datos en dos grupos: uno para entrenar el modelo (`training set`) y otro para evaluarlo (`test set`). El conjunto de entrenamiento se utiliza para que el modelo aprenda los patrones de los datos, mientras que el conjunto de prueba sirve para verificar qué tan bien el modelo generaliza ante nuevos datos no vistos durante el entrenamiento. Una división típica es de 80 % para entrenamiento y 20 % para prueba, aunque esta proporción puede variar según el tamaño y la naturaleza del conjunto de datos.
2) **Conjunto de validación (opcional)**: en muchos casos, además del conjunto de prueba, se utiliza un conjunto de validación (`validation set`), el cual permite ajustar hiperparámetros, seleccionar entre distintos modelos y prevenir el sobreajuste. Esto es especialmente útil cuando se prueba una gran cantidad de configuraciones. Una práctica común consiste en reservar un 5 % del total de los datos como conjunto de validación y utilizar el restante 15 % como prueba. En entornos dinámicos, como el financiero, la

validación puede hacerse en tiempo real, observando cómo el modelo se comporta al ser implementado.

3) **Partición estratificada**: cuando se trabaja con variables categóricas importantes, como la clase objetivo en problemas de clasificación, es recomendable emplear partición estratificada. Esta técnica asegura que la proporción de cada clase se mantenga igual en los subconjuntos (entrenamiento, validación y prueba). Esto es crucial en situaciones de desequilibrio de clases, como en el análisis de fraudes, donde los casos positivos son escasos. Ignorar esta técnica puede derivar en modelos sesgados hacia la clase mayoritaria.

Estas técnicas de partición permiten construir modelos más robustos, evitar el sobreajuste y obtener una evaluación realista del desempeño predictivo del modelo. En el siguiente apartado se abordarán estrategias de balanceo que complementan estas prácticas, especialmente cuando se enfrentan problemas de desbalance entre clases.

2.1. Métodos de partición: hold-out, k-fold y muestreo estratificado

En el contexto de la partición de datos para proyectos de aprendizaje automático, se emplean varias técnicas que permiten organizar los datos de manera eficaz para entrenar y evaluar modelos con mayor precisión (Géron, 2020). Las más utilizadas son:

1) Partición hold-out

Consiste en dividir el conjunto de datos una sola vez, generalmente en una proporción como 70 % para entrenamiento y 30 % para prueba. Esta técnica separa los datos disponibles en dos subconjuntos distintos: uno destinado al aprendizaje del modelo y otro reservado para la evaluación de su rendimiento sobre datos no vistos. La

división se realiza de forma aleatoria para evitar sesgos y asegurar que ambos subconjuntos sean representativos. Es una técnica sencilla, rápida y útil para conjuntos de datos grandes, aunque su estimación de rendimiento puede depender fuertemente de cómo se realizó la partición.

2) Muestreo estratificado

Esta técnica se utiliza cuando los datos contienen una variable categórica importante, como la clase objetivo, que debe estar representada proporcionalmente en cada subconjunto. El muestreo estratificado garantiza que cada estrato (o categoría) mantenga su proporción relativa en los conjuntos de entrenamiento y prueba. Esto es especialmente relevante en problemas de clasificación con clases desbalanceadas, donde una distribución desigual podría afectar negativamente el rendimiento del modelo.

3) Validación cruzada k-fold

Es una técnica robusta que permite evaluar modelos cuando se dispone de un volumen de datos limitado. El conjunto de datos se divide en k subconjuntos (o folds) del mismo tamaño. Luego, el modelo se entrena k veces, utilizando k–1 pliegues para entrenamiento y el pliegue restante para validación. Al rotar los pliegues, cada observación es utilizada una vez para validar y k–1 veces para entrenar. Este método reduce el riesgo de sobreajuste y proporciona una estimación más confiable del rendimiento. Comúnmente se elige k = 5 o k = 10, dependiendo del tamaño del conjunto de datos y de los recursos computacionales disponibles.

En conjunto, estas técnicas permiten aprovechar mejor los datos disponibles y obtener evaluaciones más precisas del comportamiento del modelo en condiciones reales.

2.2. Evaluación del rendimiento en conjuntos balanceados

La evaluación del rendimiento de los modelos de aprendizaje automático en conjuntos de datos equilibrados se basa en técnicas estándar de validación, aunque con la ventaja de que la distribución igualitaria entre clases permite una interpretación más clara de las métricas. Un conjunto se considera balanceado cuando contiene una cantidad similar de ejemplos para cada clase (Benítez et al., 2014), lo que reduce el sesgo en el entrenamiento y mejora la capacidad de generalización del modelo.

Una vez definida la estrategia de partición de los datos y asegurado el equilibrio entre clases, es fundamental aplicar métricas que permitan evaluar objetivamente el rendimiento del modelo. La elección de las métricas dependerá del tipo de problema (clasificación o regresión) y de los objetivos del análisis. A continuación, se describen las métricas más utilizadas en proyectos de aprendizaje automático con conjuntos de datos balanceados.

a) **División de datos**: una etapa inicial en la evaluación consiste en dividir los datos en subconjuntos para entrenamiento, validación y prueba. Es crucial que esta división mantenga la proporción de clases en cada partición, incluso si el conjunto de datos original ya está balanceado. En situaciones donde se optimizan hiperparámetros, se recomienda incluir un conjunto de validación adicional para evitar el sobreajuste durante el ajuste del modelo (Benítez et al., 2014).

b) **Métricas de evaluación**: en el contexto de conjuntos balanceados, las métricas de evaluación reflejan con mayor precisión el desempeño real del modelo, ya que no están influenciadas por clases mayoritarias. Algunas de las métricas más utilizadas incluyen:

- ✓ **Precisión (accuracy)**: mide la proporción de predicciones correctas. En conjuntos balanceados, esta métrica es confiable, a diferencia de los desequilibrados, donde puede ser engañosa.
- ✓ **Matriz de confusión**: herramienta fundamental para clasificaciones multiclase o binarias. Permite analizar verdaderos positivos (TP), verdaderos negativos (TN), falsos positivos (FP) y falsos negativos (FN), y derivar métricas clave como:
  - **Precisión (precision)**: proporción de verdaderos positivos entre las predicciones positivas.
  - E**xhaustividad o sensibilidad (recall)**: proporción de verdaderos positivos respecto al total real de la clase positiva.
  - **F1-score**: media armónica entre precisión y exhaustividad, útil cuando se requiere equilibrio entre ambas.
  - **Área bajo la curva ROC (AUC-ROC)**: indica la capacidad del modelo para discriminar entre clases. Un AUC cercano a 1.0 refleja un excelente rendimiento.
- ✓ **Métricas de regresión**: para modelos de regresión, se aplican métricas como el Error Cuadrático Medio (RMSE) y el Error Absoluto Medio (MAE), que cuantifican la diferencia entre las predicciones y los valores reales. El RMSE es más sensible a errores grandes, mientras que el MAE proporciona una visión más robusta ante valores atípicos.

c) **Validación cruzada**: especialmente la de k-fold, es una técnica esencial para estimar el rendimiento generalizado del modelo. Consiste en dividir el conjunto de datos en k pliegues, entrenar el modelo k veces utilizando diferentes combinaciones de entrenamiento y prueba, y calcular el promedio de las métricas obtenidas. Esta técnica reduce la varianza de las estimaciones y es especialmente útil con conjuntos de datos limitados. Un valor común para k es 10 (Benítez et al., 2014).

Una vez aplicadas las métricas y técnicas de validación, es recomendable comparar el desempeño del modelo con otros enfoques o algoritmos alternativos. Esta comparación permite determinar si el modelo seleccionado es el más adecuado para el problema planteado, considerando tanto su rendimiento como su capacidad de generalización. Evaluar diferentes modelos bajo las mismas condiciones facilita una toma de decisiones fundamentada y orientada a seleccionar la solución más robusta y eficiente (Raschka, 2018).

En resumen, evaluar el rendimiento en conjuntos balanceados implica mantener proporciones consistentes entre clases, aplicar métricas adecuadas para clasificación o regresión, utilizar validación cruzada para asegurar robustez y comparar con otros enfoques para validar su efectividad. El equilibrio en las clases mejora la interpretación de las métricas, aunque no modifica los principios generales de la evaluación.

3. Automatización y visualización de procesos de transformación de datos

Con el incremento en la complejidad y volumen de los proyectos de análisis de datos e inteligencia artificial, se vuelve imprescindible automatizar las tareas asociadas a la preparación y transformación de datos. La automatización no solo ahorra tiempo, sino que también garantiza consistencia, minimiza errores humanos y facilita la trazabilidad de los procesos. Esto resulta especialmente importante en entornos de producción, donde los flujos de trabajo deben ser escalables, repetibles y fácilmente mantenibles.

La implementación de pipelines o flujos de procesamiento automatizados permite encadenar diferentes etapas del preprocesamiento, como la limpieza, codificación, normalización, selección y generación de características. Estos pipelines pueden configurarse para adaptarse a nuevas entradas de datos, lo que facilita la actualización de modelos sin rehacer todo el proceso manualmente. Herramientas como Scikit-learn, TensorFlow Transform, Apache Airflow o KNIME ofrecen entornos robustos para construir y ejecutar estos flujos de trabajo de manera modular y eficiente.

Además, la visualización del proceso de transformación es clave para auditar y comprender cómo los datos cambian a lo largo del pipeline. Diagramas de flujo, gráficos interactivos y tableros de control permiten monitorear las transformaciones, identificar cuellos de botella y validar cada etapa del procesamiento. Plataformas como Power BI, Tableau, MLflow o TensorBoard brindan recursos visuales que complementan la comprensión técnica con una representación intuitiva y accesible.

En esta sección, se analizan los principios fundamentales de la automatización con pipelines, las herramientas más utilizadas para su implementación y las buenas prácticas para integrarlos en el ciclo de vida de los proyectos de aprendizaje automático. Asimismo, se destacará el valor de la visualización como un componente esencial para la supervisión y mejora continua de los procesos de transformación de datos.

3.1. ¿Qué es un pipeline de datos y para qué sirve?

Un pipeline de datos, o canal de procesamiento de datos, es una secuencia estructurada de pasos que transforma datos desde su estado bruto hasta un formato procesado y listo para ser utilizado en análisis o modelos de Inteligencia Artificial (IA). Según Géron (2020), puede entenderse como el flujo que sigue la información desde su recolección inicial hasta su utilización final como insumo en sistemas analíticos o predictivos.

Este flujo incluye múltiples etapas automatizadas que pueden integrar procesos como:

- ✓ Recolección y adquisición de datos desde fuentes diversas.
- ✓ Limpieza y depuración para eliminar inconsistencias, duplicados o valores atípicos.
- ✓ Transformación para estructurar, codificar o escalar los datos según lo requiera el modelo.
- ✓ Almacenamiento intermedio en bases de datos o estructuras temporales.
- ✓ Aplicación de algoritmos de inteligencia artificial para entrenamiento o predicción.

- ✓ Preparación y visualización de resultados, facilitando la interpretación y la toma de decisiones por parte de usuarios o clientes.

El propósito principal de un pipeline de datos es garantizar que la información fluya de manera ordenada, eficiente y reproducible desde su origen hasta su uso final, permitiendo generar valor a partir de datos crudos. Además, facilita la escalabilidad del proceso, mejora la calidad de los datos y reduce los errores manuales en tareas repetitivas.

3.2. Automatización de flujos con Scikit-learn Pipelines y MLflow

Los pipelines de Scikit-learn permiten automatizar procesos de aprendizaje automático al encadenar transformaciones de datos con un modelo estimador. Esta herramienta organiza el flujo de trabajo, facilita la reproducibilidad de los experimentos y mejora la eficiencia operativa (Géron, 2020).

En general, un pipeline se compone de varias etapas secuenciales, donde cada una aplica una transformación a los datos (como el escalado o la codificación), y la última corresponde a un estimador (clasificador o regresor). Al entrenar el pipeline, las transformaciones se ajustan automáticamente a los datos de entrenamiento; al predecir, las mismas transformaciones se aplican de forma coherente a los nuevos datos.

Una de sus principales ventajas radica en evitar errores frecuentes, como aplicar manualmente diferentes preprocesamientos al conjunto de prueba. Además, los pipelines son compatibles con herramientas de validación como GridSearchCV, permitiendo la optimización conjunta de hiperparámetros y transformaciones.

- ✓ **Ejemplo básico de uso en Python:**

```
from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

pipeline = Pipeline([

('scaler', StandardScaler()),

('model', LogisticRegression())

])
```

Este fragmento configura un `pipeline` con dos pasos:

- ✓ Estandarización de variables numéricas mediante StandardScaler.
- ✓ Entrenamiento de un modelo de regresión logística.

La ventaja es que todo el flujo puede entrenarse y reutilizarse como una unidad estructurada:

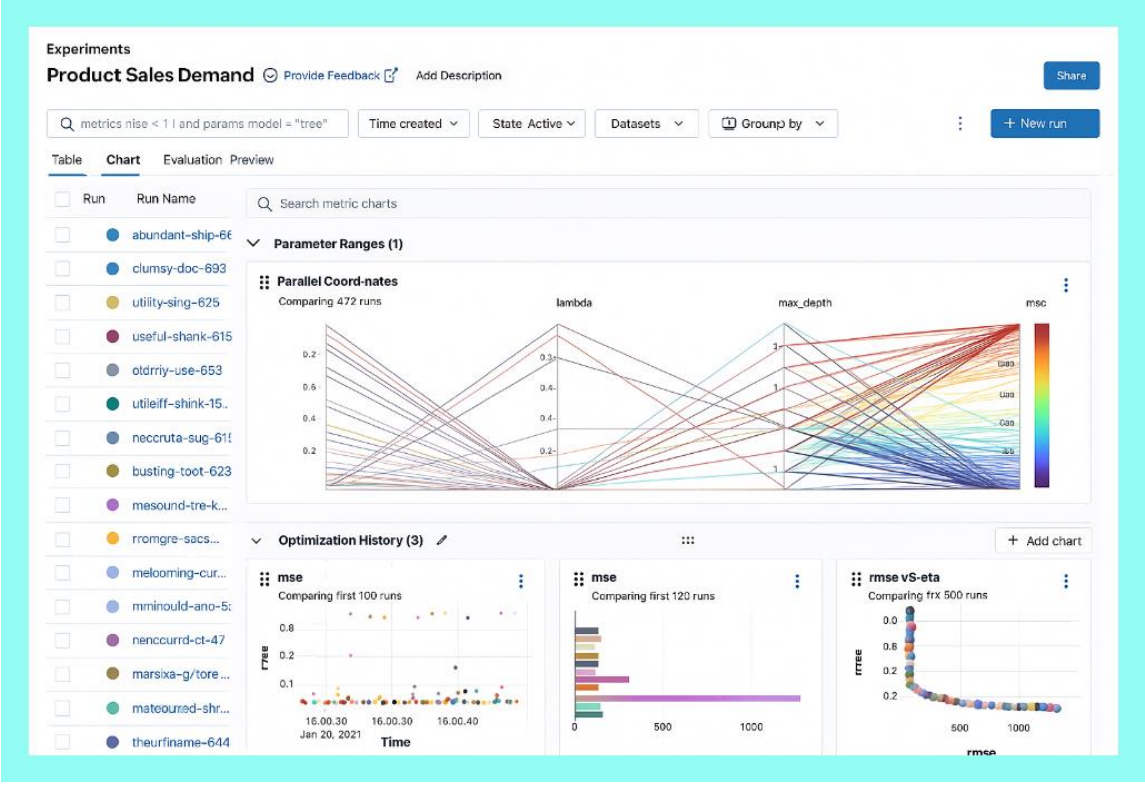
```
from sklearn.datasets import load_iris

X, y = load_iris(return_X_y=True)

pipeline.fit(X, y)

y_pred = pipeline.predict(X)
```

El código anterior demuestra un flujo completo, desde la carga del conjunto de datos hasta la predicción. Durante el proceso, se garantiza que las transformaciones se apliquen correctamente sin intervención adicional del usuario.

Por otro lado, MLflow es una plataforma de código abierto diseñada para gestionar todo el ciclo de vida del aprendizaje automático. Facilita el seguimiento de experimentos, la gestión de modelos, el registro de métricas y el despliegue en producción. Aunque no existe una integración nativa y automática entre MLflow y `Pipeline` de Scikit-learn, ambas herramientas pueden utilizarse de forma complementaria para documentar y escalar flujos de trabajo. MLflow permite:

- ✓ Registrar y versionar modelos.
- ✓ Automatizar ejecuciones y comparaciones.
- ✓ Visualizar métricas, hiperparámetros y artefactos.
- ✓ Integrarse con bibliotecas como Scikit-learn, TensorFlow o Spark.

Figura 2. Ejemplo de proyecto ML en MLflow



Nota. Tomado de https://mlflow.org/#core-concepts

La interfaz gráfica de MLflow presenta ejecuciones de modelos con diferentes combinaciones de hiperparámetros para predecir la demanda de productos. En este caso, se utiliza el RMSE (Root Mean Squared Error) como métrica principal. A través de coordenadas paralelas y gráficos de barras, es posible identificar qué configuraciones de hiperparámetros (como alpha, lambda o eta) resultan en menores errores.

Los gráficos adicionales ofrecen una visión cronológica del rendimiento, comparaciones entre modelos y relaciones entre hiperparámetros y métricas. Estas visualizaciones son esenciales para tomar decisiones informadas durante la experimentación.

Código de ejemplo con evaluación automatizada usando MLflow y modelos de lenguaje:

```
from openai import OpenAI

import mlflow

from mlflow.metrics.genai import answer_correctness, answer_similarity, faithfulness

mlflow.openai.autolog()

client = OpenAI()

prompt_template = """\

You are an expert AI assistant. Answer the user's question with clarity, accuracy, and conciseness.

## Question:
```

{question}

Guidelines:

- Keep responses factual and to the point.

- If relevant, provide examples or step-by-step instructions.

- If the question is ambiguous, clarify before answering.

Respond below:

"""

mlflow_ground_truth = (

"MLflow is an open-source platform for managing the end-to-end machine learning lifecycle..."

)

metrics = {

"answer_similarity": answer_similarity(model="openai:/gpt-4o"),

"answer_correctness": answer_correctness(model="openai:/gpt-4o"),

"faithfulness": faithfulness(model="openai:/gpt-4o"),

}

question = "What is MLflow?"

with mlflow.start_run():

response = client.chat.completions.create(

```
        messages=[{"role": "user", "content":
prompt_template.format(question=question)}],
        model="gpt-4o-mini",
        temperature=0.1,
        max_tokens=2000,
        ).choices[0].message.content
        answer_similarity_score = metrics["answer_similarity"](
        predictions=response, inputs=question, targets=mlflow_ground_truth
        ).scores[0]
        answer_correctness_score = metrics["answer_correctness"](
        predictions=response, inputs=question, targets=mlflow_ground_truth
        ).scores[0]
        faithfulness_score = metrics["faithfulness"](
        predictions=response, inputs=question, context=mlflow_ground_truth
        ).scores[0]
        logged_model = mlflow.last_logged_model()
        mlflow.log_metrics(
        {
        "answer_similarity": answer_similarity_score,
```

"answer_correctness": answer_correctness_score,

"faithfulness": faithfulness_score,

},

model_id=logged_model.model_id,

)

Con el fin de consolidar la comprensión del flujo presentado en el código anterior, a continuación, se ofrece un resumen conceptual de los principales componentes utilizados. Esta síntesis permite identificar el propósito específico de cada elemento dentro del proceso de evaluación automatizada con MLflow y OpenAI, facilitando su comprensión y posible reutilización en proyectos similares. La tabla 1 presenta de forma esquemática estos componentes y su función en el flujo de trabajo.

Tabla 1. Resumen conceptual de componentes

Componente	Propósito
mlflow.openai.autolog()	Registro automático de trazas, métricas y artefactos.
prompt_template	Controla el formato de la entrada para el modelo.
mlflow_ground_truth	Define la respuesta de referencia para la evaluación.

Componente	Propósito
metrics	Establece funciones de evaluación automática.
mlflow.start_run()	Inicia un experimento registrable.
mlflow.log_metrics()	Almacena los resultados obtenidos.
mlflow.search_traces()	Recupera información detallada de cada ejecución.

El uso conjunto de Scikit-learn y MLflow, aunque no completamente automatizado de forma nativa, permite documentar, comparar y versionar flujos de trabajo de forma profesional, haciendo que el desarrollo de modelos sea más confiable, reproducible y escalable.

3.3. Buenas prácticas para automatizar la gestión de datos

La automatización de la gestión de datos en los entornos de desarrollo de modelos de aprendizaje automático es fundamental para garantizar la eficiencia, la reproducibilidad y la escalabilidad de los proyectos. Por esta razón, es importante adoptar buenas prácticas que contemplen diversas etapas y consideraciones clave:

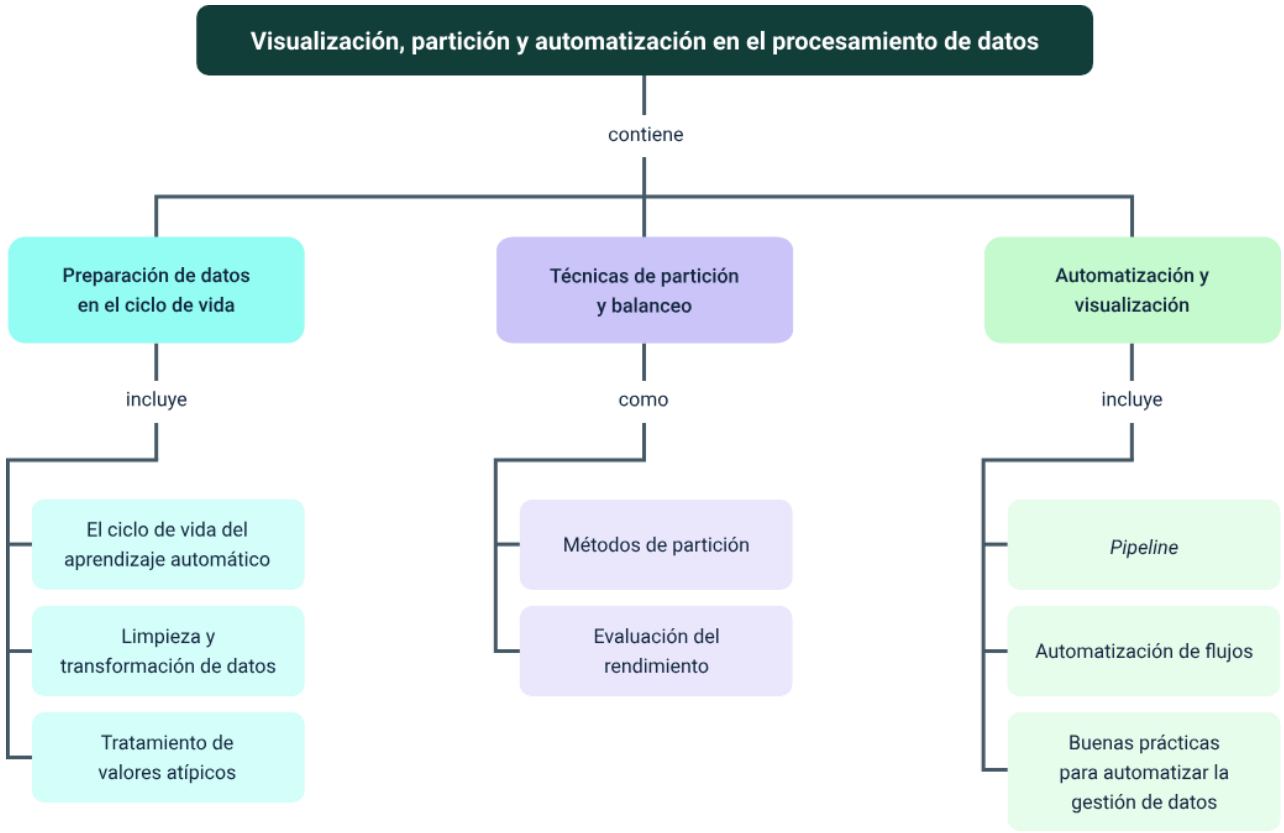
1) **Organización sistemática de los datos**: una adecuada organización facilita la interpretación y el análisis de la información. La forma más eficaz de lograrlo es mediante el uso de bases de datos, que permiten clasificaciones simples o cruzadas según las variables relevantes (Mesa Guerrero, 2020).

2) **Implementación de pipelines de datos**: un pipeline es una serie de componentes encadenados para limpiar, transformar, almacenar y aplicar algoritmos de inteligencia artificial a los datos. Un ejemplo es el pipeline de Scikit-learn, herramienta que permite estructurar un flujo de trabajo completo, automatizando desde la transformación de datos hasta la predicción, garantizando así la coherencia entre el entrenamiento y la inferencia.
3) **Preparación y transformación de datos**: este paso es esencial, especialmente en algoritmos como redes neuronales, que requieren que los datos estén normalizados en rangos específicos. Una preparación adecuada mejora el rendimiento y la estabilidad de los modelos.
4) **Automatización mediante herramientas especializadas**: el uso de soluciones ETL (Extracción, Transformación y Carga), junto con herramientas de limpieza de datos, permite automatizar el procesamiento y traslado de información desde múltiples fuentes, preparándola para su uso futuro de forma eficiente.
5) **Gobernanza de datos**: implementar un marco de gobernanza, como el propuesto por DAMA, es una práctica recomendada para garantizar el control y la integridad de la información. Esto implica establecer políticas, procedimientos, roles y responsabilidades claras, así como velar por la calidad, seguridad e interoperabilidad de los datos. En algunos casos, puede ser necesario establecer una Oficina de Datos para la supervisión de estas políticas.

En síntesis, la automatización eficaz de la gestión de datos requiere planificación estratégica, el diseño de pipelines robustos, la adopción de estándares de calidad y gobernanza, y el uso de herramientas adecuadas que se ajusten a las características del proyecto y los tipos de datos involucrados.

Síntesis

El ciclo de vida del aprendizaje automático abarca desde la recolección y limpieza de datos hasta la evaluación de modelos, destacando procesos clave como la transformación, el tratamiento de valores atípicos, la partición de datos y la automatización mediante `pipelines`. Herramientas como Scikit-learn y MLflow permiten estructurar flujos de trabajo eficientes, reproducibles y trazables. Además, la adopción de buenas prácticas y marcos de gobernanza de datos garantiza la calidad, seguridad y control de la información, contribuyendo al desarrollo de modelos más precisos y confiables.



Material Complementario

Tema	Referencia	Tipo de material	Enlace del recurso
3. Automatización y visualización de procesos de transformación de datos	Mahesh, B. (2019). Machine Learning Algorithms - A Review [Technical report]. International Journal of Science and Research (IJSR), 9(1), 381–386.	Artículo	https://www.researchgate.net/profile/Batta-Mahesh/publication/344717762_Machine_Learning_Algorithms_-_A_Review/links/5f8b2365299bf1b53e2d243a/Machine-Learning-Algorithms-A-Review.pdf?eid=5082902844932096t
3. Automatización y visualización de procesos de transformación de datos	Du, M., Liu, N., & Hu, X. (2020). Techniques for interpretable machine learning. Communications of the ACM, 63(1), 68–77.	Artículo	https://dl.acm.org/doi/pdf/10.1145/3359786

Glosario

Automatización: proceso mediante el cual se ejecutan tareas de forma automática utilizando herramientas tecnológicas, reduciendo la intervención manual y aumentando la eficiencia.

Datos atípicos: valores que se alejan significativamente del resto de los datos en un conjunto, pudiendo afectar negativamente el rendimiento de los modelos si no se tratan adecuadamente.

Evaluación del rendimiento: proceso de medir la eficacia de un modelo predictivo mediante métricas como precisión, `recall`, RMSE, entre otras.

Limpieza de datos: conjunto de técnicas para detectar y corregir errores, valores faltantes o inconsistencias en un conjunto de datos.

MLflow: plataforma de código abierto para gestionar el ciclo de vida del aprendizaje automático, incluyendo experimentación, registro de modelos y evaluación.

Modelado de datos: etapa del aprendizaje automático en la que se construyen y entrenan modelos predictivos a partir de los datos preparados.

Partición de datos: división de un conjunto de datos en subconjuntos, comúnmente entrenamiento, validación y prueba, para desarrollar y evaluar modelos.

`Pipeline`: secuencia estructurada de pasos que transforman y procesan datos, generalmente desde la limpieza hasta el entrenamiento del modelo.

Transformación de datos: proceso que modifica o convierte los datos originales para que sean adecuados para el análisis o el modelado.

Visualización: representación gráfica de datos o resultados de modelos para facilitar su comprensión, análisis y comunicación.

Referencias bibliográficas

Benítez, R., Escudero, G., Kanaan, S., & Rodó, D. M. (2014). Inteligencia artificial avanzada. Editorial UOC.

Dietterich, T. G. (1990). Machine learning. Annual Review of Computer Science, 4(1), 255–306. https://doi.org/10.1146/annurev.cs.04.060190.001351

Du, M., Liu, N., & Hu, X. (2019). Techniques for interpretable machine learning. Communications of the ACM, 63(1), 68–77.

Géron, A. (2020). Aprende machine learning con Scikit-Learn, Keras y TensorFlow: Conceptos, herramientas y técnicas para construir sistemas inteligentes. O'Reilly Media.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

Hahne, F., Huber, W., Gentleman, R., Falcon, S., & Carey, V. J. (2008). Unsupervised machine learning. En R. Gentleman, V. Carey, W. Huber, R. Irizarry & S. Dudoit (Eds.), Bioconductor case studies (pp. 137–157). Springer.

Mesa Guerrero, J. A., & Caicedo Zambrano, S. J. (2020). Introducción a la estadística descriptiva. Universidad del Valle.

MLflow. (s.f.). Deliver production-ready AI. https://mlflow.org/#core-concepts

Pyle, D. (1999). Data preparation for data mining. Morgan Kaufmann.

Python Software Foundation. (s.f.). Welcome to Python.org. https://www.python.org/

Raschka, S. (2018). Model evaluation, model selection, and algorithm selection in machine learning. arXiv.

Rudin, C., Chen, C., Chen, Z., Huang, H., Semenova, L., & Zhong, C. (2022). Interpretable machine learning: Fundamental principles and 10 grand challenges. Statistics Surveys, 16, 1–85.

Shinde, P. P., & Shah, S. (2018, August). A review of machine learning and deep learning applications. En 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA) (pp. 1–6). IEEE.

Viedma, C. D. L. P. (2018). Estadística descriptiva e inferencial. Ediciones IDT.

Créditos

Nombre	Cargo	Centro de Formación y Regional
Milady Tatiana Villamil Castellanos	Responsable Ecosistema de Recursos Educativos Digitales (RED)	Dirección General
Diana Rocío Possos Beltrán	Responsable de línea de producción	Centro de Comercio y Servicios - Regional Tolima
Deivis Eduard Ramírez Martínez	Experto temático	Centro de Comercio y Servicios - Regional Tolima
Viviana Esperanza Herrera Quiñonez	Evaluadora instruccional	Centro de Comercio y Servicios - Regional Tolima
Oscar Ivan Uribe Ortiz	Diseñador web	Centro de Comercio y Servicios - Regional Tolima
José Jaime Luis Tang Pinzón	Diseñador web	Centro de Comercio y Servicios - Regional Tolima
Veimar Celis Meléndez	Desarrollador fullstack	Centro de Comercio y Servicios - Regional Tolima
Gilberto Junior Rodríguez Rodríguez	Animador y productor audiovisual	Centro de Comercio y Servicios - Regional Tolima
Jorge Eduardo Rueda Peña	Evaluadora de contenidos inclusivos y accesibles	Centro de Comercio y Servicios - Regional Tolima

Nombre	Cargo	Centro de Formación y Regional
Jorge Bustos Gómez	Validador y vinculador de recursos educativos digitales	Centro de Comercio y Servicios - Regional Tolima